Eskişehir Osmangazi Üniversitesi Müh.Mim.Fak.Dergisi C.XXI, S.1, 2008
Eng&Arch.Fac. Eskişehir Osmangazi University, Vol. .XXI, No:1, 2008

Makalenin Geliş Tarihi : 30.07.2007
Makalenin Kabul Tarihi : 24.12.2007

GÖRÜNTÜ İSTATİSTİKLERİNDEN FAYDALANARAK LBG ALGORİTMASININ GÜNCELLENMESİ

Kemal ÖZKAN[1] , Erol SEKE[2]

ÖZET : *Veri saklama teknolojilerindeki hızlı gelişmelere rağmen görüntü verisinin saklanması ve/veya taşınması önemini yitirmemiştir. Görüntünün kodlanması ve sıkıştırılması konusundaki çalışmalar yoğun bir şekilde devam etmektedir. Yapılan çalışmalar daha fazla sıkıştırma oranı ve daha iyi bir görüntü kalitesini yakalamak içindir. Video görüntü dizilerinin kodlanması için birçok teknik geliştirilmiştir. Bu çalışmada, imgeler üzerinde başarılı sonuçlar veren vektör nicemleme yöntemi imge dizilerinin kodlanmasında kullanılmıştır. Çok bilinen vektör nicemleme algoritmalarından Linde-Buzo-Gray algoritması özyinelemeli bir süreç olduğundan büyük işlem gücü gerektirmektedir. Bu çalışmada, çerçeveler arasındaki yüksek ilişkiyi kullanarak her bir çerçeve için yeni bir kod-kitabı oluşturmak yerine ilk çerçeveden oluşturulan kod-kitabını sonraki çerçeveler için güncelleyen bir algoritma önerilmiştir. Güncelleme algoritması sayesinde yaklaşık %1 bozulmaya karşılık zamandan 3.5:1 oranında kazanç elde edilmiştir.*

ANAHTAR KELİMELER : *Veri sıkıştırma, Vektör nicemleme, LBG algoritması*

USE OF IMAGE STATISTICS FOR UPDATING LBG ALGORITHM

ABSTRACT : *Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data transmission bandwidth continues to outstrip the capabilities of available technologies. Studies on image coding and compression techniques are still going on. The objective is to achieve higher compression ratios with minimum loss of quality. Many video coding techniques have been developed. Vector quantization (VQ) is being used successfully for still images and we used VQ for coding video sequences. One of the well-known VQ algorithms is the popular Linde-Buzo-Gray algorithm. Since the LBG algorithm is an iterative process, its complexity for video sequences is very high that it requires extraordinary computational power. In this paper, we propose an algorithm that progressively updates the VQ codebooks taking advantage of high correlation between consecutive frames instead of generating a new codebook for each frame or group of frames. We have experimentally shown that with the proposed technique computational gains of 3.5:1 can be achieved at the expense of 1% distortion.*

KEYWORDS : *Data compression, Vector quantization, LBG algorithm*

[1,2] *Eskişehir Osmangazi Üniversitesi, Mühendislik Mimarlık Fakültesi,*
Elektrik-Elektronik Mühendisliği Bölümü, 26480 Batı Meşelik ESKİŞEHİR

I. GİRİŞ

Hızla ilerleyen teknolojinin daha geniş kapasiteli veri saklama üniteleri ve daha hızlı iletişim olanakları sunmasına paralel olarak, veriyi depolama ve taşıma ihtiyacı da artmaktadır. Günlük yaşamda bilgisayarların yaygınlaşması ve ihtiyaç duyulan bilgiye her koşulda ve her an erişilebilme isteği, verinin çok küçük alanlarda depolanabilmesi ve hızlı bir şekilde iletilebilmesi için bu ihtiyaca yönelik daha etkili veri sıkıştırma algoritmaları üzerinde gelecekte de araştırmanın devam edeceğinin bir işaretidir. Vektör nicemleme algoritmaları halen kullanılan ve gelecek için de umut veren veri öbekleme yöntemleridir. Kayıplı görüntü sıkıştırma tekniklerinin vazgeçilmez bir öğesi olduğu gibi, sınıflandırma algoritmaları içinde ön işlem olarak kullanıldığı uygulamalar da vardır.

Görüntülerin saklanma ve iletim ihtiyacının giderek arttığının en göz önündeki örneği sayısal televizyondur. Sayısal TV yayınlarının gerçekleştirilmesi, iletimi ve saklanması için çok büyük veri iletim hızları gerekmektedir. Şöyle ki, bir çerçevelik TV görüntüsü HDTV (yüksek tanımlı televizyon) formatında (1080p) sayısallaştırıldığı zaman $1920\text{x}1080 \approx 2\text{x}10^6$ görüntü elemanı (piksel) oluşur. Her pikselin renk/parlaklık bilgileri için 24 bit ayrıldığı düşünülürse, bir çerçeve için 1/30 saniyede $\sim 5\text{x}10^7$ bit veya bir saniyede $\sim 1.5\text{x}10^9$ bit iletilmesi gerekecektir. Buna bağlı olarak en az $3\text{x}10^9$ Hz'lik bir bant genişliğine ihtiyaç duyulduğu söylenebilir. Eğer veri sıkıştırma teknikleri kullanılmazsa normal yayın sistemlerinde bu bilgiyi iletmek olanaksızdır. Çünkü normal TV yayınları VHF ve UHF bantlarından yapılmaktadır. VHF bandı 30–300 MHz, UHF bandı 300–3000 MHz dir. Bundan dolayı bit sayısını (birim süre başına) kabul edilebilir bir seviyeye düşürmek gerekmektedir [1]. Bu indirgemeyi yaparken doğrudan çerçeve başına düşen piksel sayısının, saniyedeki çerçeve sayısının veya piksel başına düşen bit sayısının azaltılması istenmez. Çünkü amaç resmin kalitesini düşürmek değil aynı kalitedeki resmi çeşitli sıkıştırma teknikleri kullanarak daha az bit ile temsil etmektir [2]. Gerçek TV çerçeveleri içinde birbirine yakın olan pikseller arasındaki benzerlik (yada tekrarlılık) oldukça yüksektir. Bant genişliğini azaltabilmek için bu tekrarlılıktan mümkün olduğu kadar faydalanılabilir. Görüntü sıkıştırmada genel olarak bu özellik kullanılır. Ayrıca gerçek görüntü dizilerinde birbirini takip eden çerçeveler arasında da birbirlerine yakın olan pikseller arasındaki gibi benzerlik ve tekrarlılık vardır. Yani ardışıl çerçeveler arasında çok az bir değişim gözlenir.

Hareketli görüntü sıkıştırmada çerçeveler arasındaki bu tekrarlılıktan da faydalanılmaktadır.

Vektör nicemleme, depolanmak ya da iletilmek istenen verinin özelliklerini en iyi şekilde yansıtabilen daha az miktarda veri ile ifade edilmesidir. Burada, en iyi şekilde yansıtma, veri miktarını azaltırken oluşacak bozulmanın en az olduğu veri kümesini bulmaktır [3]. Vektör nicemleme kullanan bir sistem bağımsız olarak incelenebilen üç bölümden oluşmaktadır; kod-kitabı oluşturulması, kodlama ve kod çözme. Kodlanacak olan imge 2x2, 4x4, 8x8 vb. boyutta vektörlere (bloklara) bölünür. İletilecek her vektör için kod-kitabında onu en iyi temsil eden vektör (kod-vektör) bulunur ve onun sıra-numarası iletilir. Alıcı tarafında ise sıra numarası verilen kod-vektör kod-kitabından bulunup asıl vektör yerine kullanılır. Tabii ki kod-kitabının hem kodlayıcı (gönderici) hem de kod çözücüde (alıcı) bulunması gerekmektedir. Ayrıca, kod-vektörler asıl vektörler ile aynı olmadığından bir kodlama hatası yapılmış olur. Kodlama hatasının en az olabilmesi için kod-kitabındaki vektörlerin kodlanacak vektörleri en iyi şekilde temsil etmesi gerekmektedir. Bu problem ise kod-kitabı üretme işleminde ele alınır. LBG (Linda, Buzo, Gray [4]) algoritması kod-kitabı üretme aşamasında çok kullanılan yöntemlerden birisidir. Bu çalışma ile önerdiğimiz yöntem ise LBG algoritmasında ihtiyaç duyulan hesaplama gücünü azaltma amacını taşımaktadır.

II. VEKTÖR NİCEMLEME ve LBG ALGORİTMASI

Nicemlemenin amacı şu şekilde özetlenebilir: Olasılık yoğunluk fonksiyonu ile tanımlanmış bir kaynaktan üretilen işaretleri ortalamada mümkün olan en düşük bit sayısı ile kodlamak, öyle ki bu kodun çözülmesi ile elde edilen işaretler mümkün olduğunca yüksek kalitede (en-az hata) olsun. Nicemleyici tasarımında da bu iki amaç beraberce sağlanmaya çalışılır. Aynı şekilde, işaretlerin çok boyutlu olması durumunda da kaynağın olasılık yoğunluk fonksiyonu tahmin edilir ve uygun bir nicemleyici tasarlanmaya çalışılır. Eğer kaynak sayısal bir görüntü kaynağı ise çok boyutlu işaretler (vektörler) olarak görüntülerden alınmış piksel grupları (her piksel bir boyut olmak üzere) kullanılabilir. Burada vektör terimi aslında bir noktayı, vektörün uç noktasını belirtmektedir ve yazının geri kalan kısmında da aynı şekilde kullanılacaktır. Vektör nicemleme (VQ) terimi vektörlerin kümelenmesi işlemi için

de kullanılmaktadır [5]. VQ ile veri sınıflandırması ve örüntü tanıma yapılmaktadır. "En iyi" olan bir VQ şu iki özelliğe sahiptir;

1. Her vektör kendisine en yakın olan kod-vektör ile temsil edilir. Bu durumda aradaki mesafe bir hata doğurur.
2. Kod-vektörler öyle yerleştirilirler ki toplam hata (olasılık yoğunluk fonksiyonu göz önüne alınarak) en azdır.

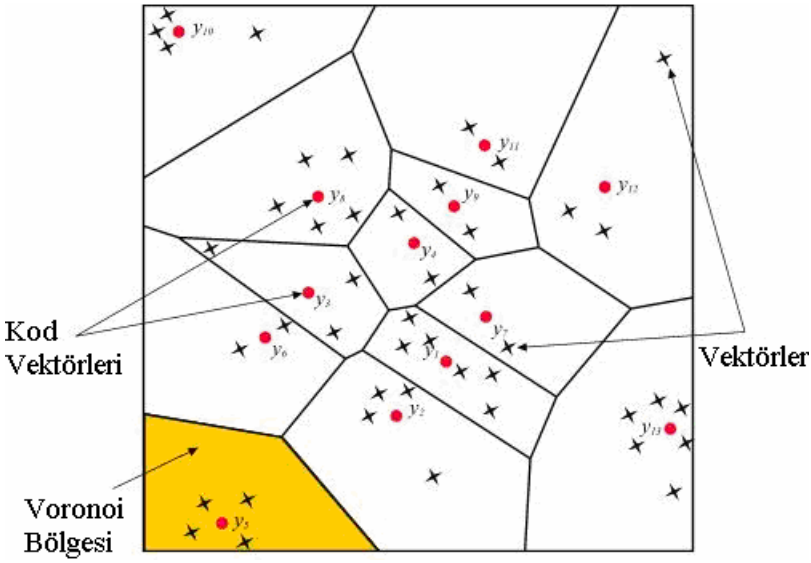
Birinci özelliğe göre vektörler S_n kodlama bölgelerinde kümelenirler ve bu küme c_n kod-vektörüne diğer kod-vektörlerden daha yakın olan tüm vektörleri içerir. Hatanın en az olması için ise c_n kod-vektörü S_n kodlama bölgesi içindeki tüm vektörlerin merkezinde olmalıdır. Bu durum, Öklid mesafeleri ile şu şekilde ifade edilebilir:

$$S_n = \left\{ x : \left\| x - c_n \right\|^2 \le \left\| x - c_{n'} \right\|^2, \forall n' = 1,2,..., N \right\} \tag{1}$$

$$c_n = \frac{\sum_{x_m \in S_n} x_m}{\sum_{x_m \in S_n} 1} \qquad n = 1,2,..., N \tag{2}$$

Şekil 1'de bir örneği gösterilmekte olan VQ bölgelerinden de görülebileceği gibi, bölgelere karşılık gelen (cluster) hücreler dışbükeydir. Bu hücrelerin içerisindeki tüm noktalar hücre merkezine (kod-vektör) diğer hücre merkezlerinden daha yakındır. Ayrıca hücrelerin birleşimi tüm vektör uzayını kapsar.

İyi bir bloklama veya vektör nicemleme için etkili algoritmalar geliştirilmiştir. En bilinen vektör nicemleme algoritmalarından LBG algoritması uzaydaki bir grup noktanın tek bir noktaya uzaklıkları toplamının en az olması için o noktanın grup merkezinde olması gerektiğini prensibi ile yola çıkar. Vektör nicemlemedeki amaç vektör uzayındaki tüm vektörleri N adet vektör ile temsil etmektir. Eldeki tüm vektörlerin uzayın olasılık dağılım fonksiyonunu tam olarak tanımlayabildiği varsayılır.



Şekil 1: Örnek VQ bölgeleri (Voronoi bölgeleri)

LBG algoritması içinde öncelikle tüm vektörleri temsil eden bir kod-vektör bulunur ve belirlenen kod-vektör sayısına ulaşıncaya kadar vektörler gruplanır. Öz-yinelemeli algoritmanın akışı ([4]) Algoritma-1'de verilmiştir.

1. Tüm vektörleri aynı gruba ata ve merkezini bul
2. Başlangıç grup merkezini ikiye böl (çok yakın iki nokta)
3. Tüm vektörleri en yakın grup merkezine ata ve grup merkezlerini bul
4. Çok sayıda vektörün grubu değiştiyse 2'ye git
5. Yeterli sayıda grup var ise dur. Merkezleri kod-vektör olarak al.
6. Tüm grup merkezlerini ikiye böl ve 3'e git.

Algoritma 1: LBG algoritması basamakları.

Bütün vektörlerin en yakın grup merkezinde (sınıfta) olması her adımdaki kod-vektörlerin veya sonuçta üretilen kod-kitabının evrensel en iyi olduğunu garanti etmez. Sonuçlar başlangıç noktasına ve kod-vektörlerinin ne şekilde ikiye bölündüğüne bağlı olarak değişir, ([6]-[8]) ancak yerel en-iyi olduğu söylenebilir.

III. GÜNCELLEME ALGORİTMASI

İmge kodlamasında kod-kitabının oluşturulmasının görüntü kalitesini belirlemede en önemli adım olduğu [6]'da Gray ve [7]'de Barlaud tarafından belirtilmiştir. Kod-kitabındaki kod-vektör sayısının artmasına bağlı olarak görüntü kalitesi artmakta fakat buna bağlı olarak sıkıştırma oranı düşmektedir. Vektör nicemleme imge dizilerinde yüksek sıkıştırma oranı sağlayabilmektedir. Aynı zamanda, basit kod çözücüler sayesinde mobil teknolojiden web uygulamalarına kadar birçok uygulamada imgelerin ve imge dizilerinin kodlanmasında verimli bir şekilde kullanma olanağı vardır. Önceki bölümlerde değinildiği üzere görüntü dizilerinin çerçeveleri arasında yüksek oranda benzerlik vardır. Yani birbirini takip eden iki çerçeve arasındaki fark çok azdır. Bundan dolayı bu iki çerçeveden elde edilen görüntü vektörleri de birbirine benzeyecektir. Barlaud [7]'de ilk ve ikinci çerçevelerden elde edilen kod-kitaplarındaki kod-vektörlerin uzaydaki yerlerinin çok az değiştiğini belirlemiştir. Önceki çerçeveler için hazırlanmış olan kod kitabı daha sonraki çerçevelerin kod-kitapları için bir başlangıç noktası olabilir. Bu çalışmada görüntü dizisinin ilk çerçevesine Algoritma-1'de verilen LBG algoritması uygulanmış, ardından gelen çerçeveler ile ilk çerçeve arasındaki benzerlikten faydalanarak tekrar kod-kitabı oluşturmak yerine ilk kod-kitabı güncellemeye çalışılmıştır. Böylelikle öz-yinelemeli algoritmanın hesaplama yükünden (ve zamandan) önemli oranlarda kazanımlar elde edilmiştir. Önerilen güncelleme algoritması Algoritma-2'te verilmiştir.

1. LBG algoritması kullanarak kod-kitabı bulunur. İlk kod kitabındaki vektörlerin yaşı sıfır yapılır.
2. Yeni eğitme vektörleri en yakın sınıfa konur. Eğitme dizisi $\{ x_j ; j = 0,1, .., n-1 \}$ olmak üzere; $x_j \in C_i$ eğer $d(x_j,y_i) < d(x_j,y_N)$
3. Her sınıftaki eğitme vektör sayısı bulunur. S_i her sınıftaki eğitme vektör sayısı olmak üzere; $s_i = s_i + 1$ eğer $x_j \in C_i$
4. Her sınıfta en az bir tane eğitme vektörü varsa durulur. Yeni kod-kitabı bulunmuş olur. $s_i = 0$ algoritma durdurulur.(bütün i'ler için) $C^{sonuç} = C$ olur.
5. Her sınıftaki en uzak eğitme vektörleri bulunur. $z_i = x_j$, $\max(d(x_j,y_i) \in C_i$ z_i vektörleri uzaklık sırasına göre büyükten küçüğe sıralanır.
6. Boş eğitme vektörüne sahip sınıf sayısı bulunur. k : boş eğitme vektörüne sahip sınıf sayısı k= k+1 eğer $s_i = 0$ bütün i'ler için $v_i = z_i$, $0 \le i \le k$
7. k sayısı kadar, en uzak vektör kendi sınıflarından çıkartılır. Çıkartılan her vektör yeni bir sınıf oluşturulur.

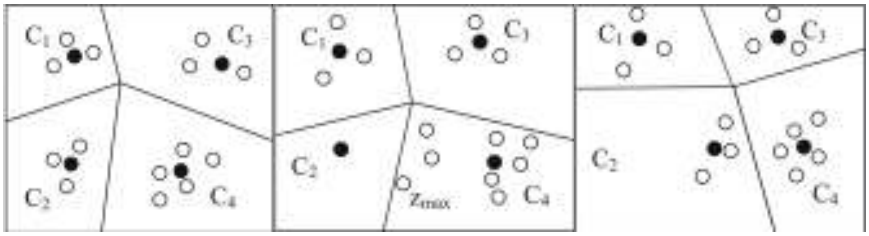
   $C_i = C_i$ eğer $s_i \neq 0$

   $C_i = C_i - v_k$ eğer $s_i \neq 0$, $0 \le i \le N - k$ ve

   $C_{i+k} = v_k$ $i < k \le N$
8. Her sınıfın ağırlık merkezi hesaplanır. Bu ağırlık merkezi o sınıfı temsil edecek kod-vektörüdür.
9. Eklenen her yeni vektörün yaşı sıfır olmak üzere vektörlerin yaşları 1 artırılır. Yaşlanan vektörlerin kod kitabındaki ağırlıkları yaşla ters orantılı olarak azaltılır.

   $$Ağırlık = e^{yaş / toplam\ çerçeve\ sayısı}$$
10. 2'ye git.

Algoritma 2: Önerilen güncelleme algoritması basamakları.



(a) (b) (c)

Şekil 2. Önerilen yöntemin uygulandığı bir dağılım. a) Başlangıçtaki hücreler ve hücre merkezleri (LBG ile bulunmuş) b) Yeni bir vektör setinin başlangıçtaki gruplara göre konumu. c) Yeniden gruplama ile elde edilen hücreler ve merkezleri.

Önerilen algoritma için örnek olarak verilen 14 adet vektörün dağılımı ve başlangıçtaki gruplaması (LBG ile bulunmuş) Şekil 2a'daki gibi olsun. Burada C_i ler hücreleri, içi dolu yuvarlaklar ise hücre ağırlık merkezini göstermektedir. Şekil 2b'de ise bir sonraki veri dizisinin (sonraki çerçeve) ilk kod-kitabına göre dağılımı gösterilmiştir. Şekil 2'de görüleceği gibi C_2 hücresine hiç bir veri düşmemiş ve kod-kitabı büyüklüğümüz 4 yerine 3 olmuştur. Daha sonra, tüm vektörler içinde kendi hücre merkezine göre en uzak olan vektör bulunmuş (Şekil 2b'de $z_{mak.}$ ile gösterilen) ve bu vektör yeni bir hücrenin elemanı olarak kabul edilmiştir. Veri setimiz yeni yapıya göre Algoritma-2'de verilen 3'üncü adımdan başlamak üzere tekrar gruplandırılmıştır.

IV. UYGULAMA

Hareketli görüntüler için 39 çerçeveden oluşan *tenis*, 60 çerçeveden oluşan *amerikan futbolu*, ve 99 çerçeveden oluşan *bahçe* görüntüleri kullanılmıştır. Üç görüntü dizisinin her çerçevesine LBG algoritması uygulanmış ve kod-kitabı büyüklükleri olarak da 2, 4, 8, 16, 32, 64, 128, 512, 1024 seçilmiştir. Güncelleme algoritmasında ise her üç görüntü dizisinin ilk çerçevesinden elde edilen kod-kitabı kullanılarak sonraki kod-kitapları bulunmuştur. [11] deki tablolar, LBG algoritması ile karşılaştırabilmesi için sinyal gürültü oranları tablo halinde tekrar verilmiştir. Futbol dizisinin 15'inci çerçevesi için elde edilen örnek sıkıştırılmış-çözülmüş görüntüler Şekil 3 de verilmiştir. En iyi sinyal-gürültü oranı doğal olarak kod-kitabı büyüklüğü 1024 olduğu zaman bulunmuştur. Kod-kitabı büyüklüğü artırıldığı zaman SNR artmakta, ancak sıkıştırma oranı azalmaktadır. Ayrıca kod-kitabı büyüklüğü ile gerçekleme zamanı arasında ters orantı vardır [11]. Kod-kitabı büyüklüğü 1024 olduğu zaman LBG algoritması ve güncelleme algoritması sonucu elde edilen SNR değerleri Tablo 1'de verilmiştir. Tablo 1'de yaşlandırma ile vektörlerin ağırlıkların azaltılmasının etkisi son sütün da görülmektedir. Tablo 2'de ise algoritmaların gerçekleme zamanları saniye cinsinden verilmiştir. Verilen süreler gerçek zaman uygulamaları için uygun olmadığından burada yapılanın bir karşılaştırma olduğunun ve sürelerin bilgisayar gücüne ve yazılıma bağlı olduğunun hatırlatılmasında yarar vardır. Güncelleme algoritması kullanıldığı zaman amerikan futbolu dizisi için ortalama olarak 1.19 db (%0.96), tenis dizisi için 0.52 db (%0.49) ve bahçe dizisi içinde 1.16 db (%1.21) bozulma olmuştur. Bu bozulma miktarlarına karşın her

çerçeve için ortalama olarak sırasıyla 668, 602 ve 534 sn kazanılmıştır. Yani gerçekleme zamanı olarak futbol dizisi için 3.61:1, tenis dizisi için 3.78:1 ve bahçe dizisi için 3.39:1 oranında zamandan kazanım olmuştur [12].

Tablo 1. Görüntü dizilerinde bulunan ortalama SNR

Ortalama	Sinyal Gürültü Oranı (SNR) (db)			
	Her Çerçeveye	İlk Çerçeveye	Güncelle Algoritması	Yaşlandırmalı Algoritması
Futbol	122.78	93.37	121.59	122.18
Tenis	104.83	90.66	104.32	104.59
Bahçe	96.18	81.12	95.00	95.78

Tablo 2. Görüntü dizilerinin gerçekleme zamanı

	Gerçekleme Zamanı (sn)			
	Toplam Zaman		Ortalama Zaman	
	LBG	Güncelleme	LBG	Güncelleme
Futbol	55391	15327	923.18	255.45
Tenis	31937	8451	818.90	216.69
Bahçe	75030	22138	757.88	223.62

V. SONUÇLAR

Önerilen güncelleme algoritması ile yüksek sıkıştırma oranı elde edilebilen (11:512) vektör nicemlemede en çok kullanılan LBG algortimasıın en büyük problemi olan gerçekleme zamanı düşük bozulma oranları ile birlikte azaltılabilmiştir. LBG algoritması gruplar arasındaki ilintiyi azaltır, ancak resim üzerindeki bloklar arasındaki ilintiyi dikkate almaz. Dolayısıyla tekrar oluşturulan görüntülerde blok etkileri belirgin olarak görülmektedir. Blok etkileri çeşitli filtrelerle azaltılabilmekte, ancak bu sadece görüntünün izlenebilirliğini artırmakta, sinyal gürültü oranı ise azalmaktadır. LBG algoritmasının bir başka etkisi de yumuşatma etkisidir. Bundan sonraki çalışmalar bahsedilen etkilerin giderilmesi üzerine olacaktır.

(a)

(b)

(c)

Şekil 3. **a)** Orijinal futbol resmi (15.çerçeve)

b) Klasik LBG algoritması

c) Yaşlandırma algoritması.

KAYNAKLAR

[1] Umbaugh, S. E., Computer Vision and Image Processing, Prentice-Hall International Inc., 1998.

[2] Gonzalez, R., Woods R., Digital Image Processing, Addision-Wesley Publishing Company, 1992.

[3] Duda R. O., Hart P. E. ve Stork D.G., Pattern Classification 2nd Edition, Wiley-Interscience Publication, USA.2001.

[4] Y. Linde, A. Buzo, & R. M. Gray, " An Algorithm for Vector Quantization Design ," IEEE Transactions on Communication, v. COM-208, pp.84-95, 1980.

[5] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic, 1992.

[6] Gray R., Vector Quantization, IEEE ASSP Magazin, April, 1984.

[7] Barlaud M., Pyramadial Lattice Vector Quantization for Multiscale Image Coding., IEEE Trans. on Image Processing, Vol.3, No.4, July, 1994.

[8] Nasrabadi N., Feng Y., Image Compression Using Address-Vector Quantization, IEEE Trans. on Communications, Vol.38, No.12, December, 1980.

[9] Nasrabadi N., King R., Image Coding Using Vector Quantization: A Review, IEEE Trans. on Communications, Vol 36., No.8, August, 1998.

[10] Nasrabadi N., Feng Y., Image Compression Using Address-Vector Quantization, IEEE Trans. on Communications, Vol.38, No.12, December, 1980.

[11] K. Özkan, E. Seke, LBG Algoritmasının Görüntü Dizeleri İçin Güncellenmesi, Elektrik-Elektronik ve Bilgisayar Sempozyumu, ELECO 2006, 6 –10 Aralık 2006, Bursa

[12] K. Özkan, LBG Algoritmasında Görüntü İstatistiklerinden Faydalanma, Y. Lisans Tezi, Osmangazi Üniversitesi, Eskişehir, Temmuz 2000